Exhibit I

To

Joint Claim Chart

08/99/277

ABSTRACT

A method and system for receiving user input data into a computer system having a graphical windowing environment. A touch-sensitive display screen for displaying images and detecting user activity is provided. A management component connects to the graphical windowing environment to create an input panel window for display on the screen. An input method which may be a COM object is selected from multiple input methods available, and installed such that the input method can call functions of the management component. Each input method includes a corresponding input panel, such as a keyboard, which it draws in the input panel window. When the user taps the screen at the input panel, the input method calls a function of the management component to pass corresponding input information appropriate information such as a keystroke or character to the management component. In response, the management component communicates the user data to the graphical windowing environment as a message, whereby an application program receives the message as if the message was generated on a hardware input device.

S P E C I F I C A T I O N

TO ALL WHOM IT MAY CONCERN:

Be it known that we, Michael G. Toepke, a citizen of the United States, residing at 14707 NE 36th #A7, Bellevue, Washington 98007, Jeffrey R. Blum, a citizen of the United States, residing at 231 Belmont Ave. E. #203 Seattle, Washington 98102, and Kathryn L. Parker, a citizen of the United States, residing at 35539 SE 41st Street, Fall City, Washington 98024, have invented a certain new and useful SOFT INPUT PANEL SYSTEM AND METHOD of which the following is a specification.

SOFT INPUT PANEL SYSTEM AND METHOD

FIELD OF THE INVENTION

The invention relates generally to computer systems, and more particularly to the input of data into a computer system.

BACKGROUND OF THE INVENTION

Small, mobile computing devices such as personal desktop assistants including hand-held and palm-top computers and the like are becoming important and popular user tools. In general, they are becoming small enough to be extremely convenient while consuming less and less battery power, and at the same time becoming capable of running more and more powerful applications.

Although such devices continue to shrink in size, size limitations are being reached as a result of human limitations. For example, a full character keyboard that enables user data input cannot be so small that human fingers cannot depress the individual keys thereon. As a result, the size of such devices (e.g., palm-top computers) has become limited to that which can accommodate a full character keyboard for an average user.

One solution to reducing the size of the portion of the device that receives user input is to provide a touch-sensitive display, and thereby substantially eliminate the

need for a physical keyboard. To this end, an application program such as a word processor displays a keyboard, whereby the user enters characters by touching the screen at locations corresponding to the displayed keys. Of course, touch screen devices can also be used simultaneously with devices having a physical keyboard, whereby characters can also be entered by manually pressing the keys of the physical keyboard.

While a touch-screen device serves to provide a suitable means of user data entry, the data entry panel is typically part of the application program, i.e., each application needs to develop its own touch-sensitive interface. As a result, a substantial amount of duplication takes place. For example, both the word processor and a spreadsheet program require alphanumeric keyboard input, whereby each provides its own touch-screen keyboard interface. Other types of programs, such as a calculator program, need a numeric keypad with additional keys representing mathematical operations. This makes each program larger, more complex and consumes computer system resources.

Alternatively, the operating system can supply all the virtual keyboards and thus eliminate the redundancy, however this limits applications to using only those virtual keyboards supplied by the operating system. Newer applications (e.g., those added by plug-in modules) are unable to provide an input

mechanism that is more tailored to its particular needs. For example, a new paintbrush program may need its own graphical input screen. In sum, there is a tradeoff between flexibility and efficiency that is inherent with present user data input mechanisms.

OBJECTS AND SUMMARY OF THE INVENTION

Accordingly, it is an object of the present invention to provide an improved method system for entering user data into a computer system.

Another object of the present invention is to provide the method and system for user data entry that is both efficient and flexible.

In accomplishing those objects, it is a related object to provide a method and system of the above kind that functions with touch-sensitive input mechanisms.

Yet another object is to provide a method and system as characterized above that enables a plurality of applications to receive user input from a common input method.

A related object is to provide a method and system that enables selection of one or more input methods for each application from among a set of interchangeable input methods.

Yet another object is to provide such a method and system that is cost-effective, reliable, extensible and simple to implement.

Briefly, the present invention provides a method and system for receiving user data input into a computer system, such as a computer system having a graphical windowing environment. The invention may utilize a touch-sensitive display screen for displaying images and detecting user contact therewith (or proximity thereto). A management component operatively connected to the graphical windowing environment creates an input panel window for display on the screen. An input method is selected from among a plurality of such input methods and installed, whereby the input method can call functions of the management component. Each input method includes a corresponding input panel, such as a keyboard, which it draws in the input panel window. When user data is received via the input panel, the input method calls a function of the management component to pass the user data thereto, and in response, the management component communicates the user data to the graphical windowing environment such as in a windows message. An application program receives the message, such as corresponding to a keystroke, as if the message was generated on a hardware keyboard.

Other objects and advantages will become apparent from the following detailed description when taken in conjunction with the drawings, in which:

BRIEF DESCRIPTION OF THE DRAWINGS

FIGURE 1 is a block diagram representing a computer system into which the present invention may be incorporated;

FIG. 2 is a block diagram representing various components and connections therebetween for implementing interchangeable input panels according to an aspect of the present invention;

FIG. 3 is a flow diagram generally representing a process for getting user input from a selected input method to a selected application in accordance with one aspect of the present invention;

FIG. 4 is a state diagram generally representing SIP selection states;

FIG. 5 represents a display on a touch-sensitive display screen on an exemplary computing device;

FIG. 6 represents a display on a touch-sensitive display screen on an exemplary computing device providing the ability to select from among interchangeable input panels in accordance with the present invention;

FIG. 7 represents a display on a touch-sensitive display screen wherein a keyboard has been selected as an input panel in accordance with the present invention; and

FIG. 8 is a flow diagram representing the general steps taken in response to a change in SIP status.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Exemplary Operating Environment

Figure 1 and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the invention may be implemented. Although not required, the invention will be described in the general context of computer-executable instructions, such as program modules, being executed by a hand-held computing device such as a personal desktop assistant. Generally, program modules include routines, programs, objects, components, data structures and the like that perform particular tasks or implement particular abstract data types.

Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including palm-top, desktop or laptop personal computers, mobile devices such as pagers and telephones, multi-processor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe

computers and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

With reference to FIG. 1, an exemplary system for implementing the invention includes a general purpose computing device in the form of a hand-held personal computing device 20 or the like, including a processing unit 21, a system memory 22, and a system bus 23 that couples various system components including the system memory to the processing unit 21. The system bus 23 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. The system memory includes read-only memory (ROM) 24 and random access memory (RAM) 25. A basic input/output system 26 (BIOS), containing the basic routines that help to transfer information between elements within the hand-held computer 20, such as during start-up, is stored in the ROM 24.

A number of program modules are stored in the ROM 24 and/or RAM 25, including an operating system 28 (preferably Windows CE), one or more application programs 29, other

program modules 30 and program data 31. A user may enter commands and information into the hand-held computer 20 through input devices such as a touch-sensitive display screen 32 with suitable input detection circuitry 33. Other input devices may include a microphone 34 connected through a suitable audio interface 35 and a physical (hardware) keyboard 36 (FIG. 2). The output circuitry of the touch-sensitive display 32 is also connected to the system bus 23 via video driving circuitry 37. In addition to the display 32, the device may include other peripheral output devices, such as at least one speaker 38 and printers (not shown).

Other external input or output devices 39 such as a joystick, game pad, satellite dish, scanner or the like may be connected to the processing unit 21 through an RS-232 or the like serial port 40 and serial port interface 41 that is coupled to the system bus 23, but may be connected by other interfaces, such as a parallel port, game port or universal serial bus (USB). The hand-held device 20 may further include or be capable of connecting to a flash card memory (not shown) through an appropriate connection port (e.g., slot) 42 and interface 43. A number of hardware buttons 44 such as switches, buttons (e.g., for switching application) and the like may be further provided to facilitate user operation of the device 20, and are also connected to the system via a

suitable interface 45. An infrared port 46 and corresponding interface/driver 47 are provided to facilitate communication with other peripheral devices, including other computers, printers, and so on (not shown). It will be appreciated that the various components and connections shown are exemplary and other components and means of establishing communications links may be used.

SOFT INPUT PANEL

The soft input panel architecture is primarily designed to enable character, key-based and other user data input via the touch screen 32 of the device 20 rather than a physical keyboard 36. However, as can be appreciated, a given computer system 20 may optionally and additionally include a physical keyboard, as represented by the dashed box 36 of FIG. 2. Moreover, as will become apparent, the "soft input panel" need not be an actual touch-sensitive panel arranged for directly receiving input, but may alternatively operate via another input device such as the microphone 34. For example, spoken words may be received at the microphone 34, recognized, and displayed as text in an on-screen window, i.e., a soft input panel.

FIG. 2 shows a block diagram implementing the SIP architecture in accordance with one aspect of the present

invention. The computer system 20 includes an operating system (OS) 28 such as the graphical windowing environment 60. Such a graphical windowing environment 60 is generally operational to receive user input through a variety of devices including the keyboard 36, a mouse (not shown), a digitizer (not shown) and so on. In turn, the graphical windowing environment 60 may provide such user input to an application having "input focus," typically in the form of a keyboard character event. Note that a number of applications 29 may be executable by the computer system, however one application that is currently running is said to have "input focus" and receive the input.

In accordance with one aspect of the present invention, the present architecture employs a SIP manager 58 to provide a single and flexible interface for a plurality of different input methods 64. In general, the SIP manager 58 provides keystrokes from a selected input method 64 to the graphical windowing environment 60 (e.g., the Windows CE operating system 28). Once received, the graphical windowing environment 60 sends information corresponding to the user input data to an application 29 (i.e., the application whose window currently has input focus) in the form of that keystroke, mouse or other message placed in the message queue of the application's window. The passing of such messages is

well known in Windows programming and is described in *"Programming Windows 95,"* Charles Petzold, Microsoft Press (1996), hereby incorporated by reference. As a result, any application capable of handling keyboard input may be used with any appropriately-configured input method 64. Indeed, if an optional keyboard 36 is present, keystrokes are directly provided by a keyboard driver 62 to the graphical windowing environment 60, whereby appropriate keystrokes are likewise placed in the message queue of the active application's window without the application being provided with information as to the source.

Input methods 64 may include, for example, various different displayable keyboards, (soft keyboards), a calculator, a formula and/or equation editor, chemical symbol template, voice recognition, handwriting recognition, shorthand symbol recognition (such as "Graffiti"), or other application-optimized input methods (e.g. a barcode reader). The SIP manager 58 provides a user interface for permitting a user to toggle a SIP window (panel) 50 (FIG. 7) between an opened and closed state, as described in more detail below. The SIP manager 58 also provides a user interface enabling user selection from a displayable list of available input methods. A user interacting with the user interface may select an input method 64, and in response, the SIP manager 58

loads and calls the selected input method 64. In a preferred embodiment, each of the input methods communicates with the SIP manager 58 through a COM (Component Object Model) interface shown as IIMCallback 61 and IInputmethod 63. A COM object comprises a data structure having encapsulated methods and data that are accessible through specifically defined interfaces. A detailed description of COM objects is provided in the reference entitled "Inside OLE," second edition, Kraig Brockschmidt (Microsoft Press), hereby incorporated by reference.

Generally, when the SIP window 50 is toggled between on/off by a user, as will be described in more detail below, the SIP manager 58 informs the selected input method 64 to correspondingly open/close the SIP window 50 through the IInputmethod mechanism 63. When a new input method is selected, the SIP manager 58, through the mechanism 63, informs any of the previously selected input methods to exit, and loads the newly selected input method. The ~~interface~~ mechanism 63 may also be utilized by the SIP manager 58 to obtain information specific to a selected input method, as also described in detail below.

The selected input method 64 may also communicate information to the SIP manager 58 via the IIMCallback mechanism 61, such as which character or characters were

entered by a user, irrespective of whether the character or characters are generated through keyboard selection, handwriting recognition, voice recognition, a formula editor, calculator or the like. Such character input is generally passed to the SIP manager 58, preferably received as (or converted to) a Unicode character (for Windows CE) by the SIP manager 58 and output to the graphical windowing environment 60. Command key information, such as "Ctrl" on a keyboard, may also be provided by the input method 64 to the SIP manager 58 via interface 61.

SIP and input method-specific information may also be communicated through the SIP manager 58, and ultimately to the focused application 29, when the application is optimized for operating with a SIP (i.e., is "SIP-aware") as described in more detail below.

The system operates as generally represented in the steps of FIG. 3. Once an application is selected and has focus (steps 300 - 302), an input method 64 is selected therefor at step 304. Note that the input method 64 may be selected by the user, or a default input method may be selected for use with a particular application. Additionally, the input method 64 may be one that remains after having been selected for a previous application, i.e., a particular input method stays the same as the user switches between various applications.

In any event, the input method 64 displays a SIP window 50 when selected.

As the user inputs data at step 306, appropriate data is passed to the SIP manager 58 via the IIMCallback mechanism 61, described below. Note that the input method 64 may first process the received data at step 306. By way of example, one particular input method 64 may convert barcode symbols to Unicode characters representing digits, another input method may convert mathematical entries into a Unicode result (e.g., an entry of '3+6=' sends a '9' to the SIP manager 58), while yet another may be an equation editor (e.g., the characters "Sqrt" are converted into a single Unicode value representing a square root symbol). After any such processing, the input method 64 passes those digits to the SIP manager 58, which in turn passes those digits to the graphical windowing environment 60. The application receives the character data from the graphical windowing environment 60 as if the user had entered those digits on a physical keyboard, regardless of the input method used.

As shown in FIGS. 5-7, the soft input panel (SIP) functionality of the system collectively includes the visible window 50 (FIG. 7), a visible SIP button 52, and various methods and functions (described below). As shown in FIG. 7, the SIP window 50 is a rectangular area provided by the input

method 64 that can be hidden or shown at the user's (or an application program's) request. The visible SIP button 52 is located on a taskbar 56 or the like, and provides a touch-sensitive interface by which the user displays or hides the SIP window 50. Thus, as represented in the state diagram of FIG. 4, the window 50 toggles between an open, visible state (FIG. 7) and a closed, hidden state (FIG. 5) as the user taps the SIP button 52. A present design implements a 240 pixel wide by 80 pixel high SIP window 50 that is fixed (docked) on the display 32 at a position just above the taskbar 56. As will become apparent below, the soft input panel design supports other SIP window 50 sizes or positions.

To this end, the operating system 28 creates a dedicated thread (the SIP manager 58) that registers itself as a SIP thread with the Windows CE system. The thread creates the SIP window 50, performs other SIP initialization, and then enters a message loop to respond to messages and user interface activity in the SIP window 50. The thread also serves to dispatch messages to an Input Method's window, and calls into the Input Method 64 to permit the Input Method 64 to create windows that will respond as special SIP windows.

The SIP manager thread 58 is given special status by the system. For example, windows created by the SIP manager 58 thread are topmost windows, and ordinarily will not be

obscured by other windows, except, e.g., when the taskbar 56 is activated in an auto-hide mode while the SIP window 50 is displayed. In this case, the SIP window 50 remains displayed in its current location and the taskbar 56 is displayed on top of the SIP window 50. More generally, any user interface element for controlling the SIP may (and should) be placed on top of (rather than underneath) the SIP window 50, whenever the controlling user interface element and the SIP window 50 overlap.

Moreover, when tapped on, the SIP window 50 (and any child windows thereof such as pushbuttons, text entry fields, scrollbars and the like) will not receive the input focus as would conventional program windows. In this manner, the user may interact with the SIP window 50 without changing the system focus. As can be appreciated, changing the system focus each time the user inputs data into the SIP window 50 would be undesirable. The SIP button 52 will also not cause a change of focus for the same reason, i.e., it is undesirable to cause the window with focus to lose focus by tapping on the SIP button 52 to bring out the SIP window 50.

In accordance with one aspect of the present invention, the SIP system enables the selective installation of a specified Input Method 64. As generally described above, each Input Method 64 is an interchangeable component by which the

user provides character, text or other user data via the touch-screen display (or some other input device). More particularly, the SIP manager 58 preferably exposes a COM interface that enables the selective installation of Input Methods 64. The Input Method 64 occupies space inside a SIP window 50 created by the system.

Preferably, the Input Method 64 comprises a Component Object Model (COM) object that implements the IInputMethod interface. Notwithstanding, the Input Method 64 and SIP manager 58 can comprise virtually any components capable of communicating with one other through some mechanism, such as by receiving, responding to, and making function calls.

The Input Method 64 is responsible for drawing in the SIP window 50 and responding to user input in the SIP window 50. Typically, the Input Method 64 will respond to user input and convert that input into characters which are then sent to the SIP manager 58 via exposed SIP functions. By way of example, one Input Method 64 includes a default QWERTY (alpha) keyboard 66 shown in FIG. 7. More particularly, this Input Method 64 displays an image of the keyboard 66 on the screen 32, and converts taps on that keyboard 66 (detected as screen coordinates) into characters which are sent to the SIP manager 58 and thereby to the system. Input Methods may be written by

application vendors, and are added to the system using COM component installation procedures.

The user interacts with the Input Method 64 manifested in the visible SIP window 50 to create system input. As best represented by the state diagram of FIG. 4 and as shown in FIG. 6, the user can select a different Input Method by tapping a SIP menu button 70 on the taskbar 56 that provides a pop-up input method list 72 into the SIP window 50. The user can also select among available Input Methods via a control panel applet (not shown) or the like. The SIP control panel applets communicate with the operating system 28 using the registry and the exposed SIP-aware functionality described below.

As will be described in detail below, the various components cooperate to expose functions, structures, and window messages that enable system applications 29 to respond to changes in the SIP state. An application 29 that uses this functionality to adjust itself appropriately to SIP changes is considered "SIP-aware." Other applications may be SIP-aware yet choose to retain their original size (and thus be partially obscured by the SIP window 50) when appropriate. Moreover, and as also described below, there are exposed functions that enable applications to programmatically alter the SIP state.

Notwithstanding, applications 29 need not be aware of the SIP system in order to benefit from the present invention. Indeed, one aspect of the present invention is that applications do not ordinarily recognize whether data received thereby originated at a hardware input device such as the keyboard 36 or via user activity (e.g., contact or proximity detected by the screen 32 and detection circuitry 33) within the soft input panel window 50. This enables applications to operate with virtually any appropriate input method, irrespective of whether that application is SIP-aware.

Turning to an explanation of the mechanism that facilitates the operation of an Input Method 64 installed by the SIP manager 58, a SIP-aware application 29 is notified when the SIP window 50 changes state and what the new, current state of the SIP window 50 is. The state includes whether the status of the SIP window 50 is visible or hidden, whether the SIP window 50 is docked or in a floating condition, and the size and position of the SIP window 50. As shown in the table below, a data structure (SIPINFO) contains this SIP information:

```
Typedef struct {
     DWORD  cbSize
     DWORD  fdwFlags
     RECT   rcVisibleDesktop
     RECT   rcSipRect
     DWORD  dwImDataSize
     Void *pvImData
} SIPINFO;
```

The cbSize field may be filled in by the application program 29 and indicates the size of the SIPINFO structure. This field allows for future enhancements while still maintaining backward compatibility, and indeed, the size of the SIPINFO structure may be used to indicate the version to the components of the system. The fdwFlags field represents the state information of the SIP window 50, and can be a combination of three flags. A SIPF_ON flag that is set indicates that the SIP window 50 is visible (i.e., not hidden), while a set SIPF_DOC flag indicates the SIP window 50 is docked (i.e. not floating). A set SIPF_LOCKED flag indicates that the SIP window 50 is locked, i.e., the user cannot change its visible or hidden status. Note that a given implementation may not allow floating or locked SIP windows, however the capability is present within the system.

The rcVisibleDesktop field contains a rectangle, in screen coordinates, representing the area of the screen desktop 68 not obscured by the SIP window 50. If the SIP

window 50 is floating (not docked), this rectangle is equivalent to the user-working area. Full-screen applications wishing to respond to SIP window 50 size changes can generally set their window rectangle data structure ("rect") values to this RECT data structure's values. If the SIP window 50 is docked and does not occupy an entire edge (top, bottom, left or right), then this rectangle represents the largest rectangle not obscured by the SIP window 50. However, the system may provide available desktop space 68 not included in the RECT data structure.

Next, the rcSipRect field contains the rectangle, in screen coordinates, representing the size and location of the SIP Window 50. Applications 29 will generally not use this information, unless an application 29 wants to wrap around a floating SIP window 50 or a docked SIP window 50 that is not occupying an entire edge.

The dwImDataSize field contains the size of the data pointed to by the PvImData member, which is the next field, i.e., a pointer to the Input Method-specific data. The data are defined by the Input Method 64.

Whenever the state of the SIP window 50 changes, i.e., a new Input Method has been selected and/or a visibility, docking or size change has occurred, a message, WM_SETTINGCHANGE, is sent to all top-level windows, as

generally represented at step 800 of FIG. 8. In this manner, an application 29 can adjust itself to the new state of the SIP window 50, such as by adjusting its size in response to this message. To this end, a flag, SPI_SETSIPINFO, is sent with this message to indicate when SIP information has changed, and another flag, SPI_SETCURRENTIM, when the current Input Method has changed. As shown at step 802 of FIG. 8, the flag is tested to determine if the message is SIP-related or another type of setting change message (whereby it is handled at step 804). If SIP-related, for performance reasons, the applications that are not currently active in the foreground cache these SIP changes (steps 806 - 808). If the application's window is active, the application can adjust its size and/or window (steps 810 - 812). For example, as shown in FIGS. 5 and 6, when the SIP window 50 of FIG. 7 is hidden and an active application 29 notified, the application 29 may use the additional desktop space 68 to display more information such as the analog clock faces. Note that an application 29 that has cached a SIP change when inactive can query the current SIP state when activated to subsequently adjust itself in an appropriate manner in accordance with the information that is returned.

To query the SIP manager 58, another function, SHSipInfo, is provided so that applications 29 can determine information

about the SIP window 50 and Input Method 64. In general, if this function succeeds, the return value will be nonzero, while if this function fails, the return value will equal zero and extended error information will be available via a GetLastError() call.

The following table sets forth the structure of this call:

```
SHSipInfo(
     UINT uiAction
     UINT uiParam
     PVOID pvParam
     UINT fwinIni
);
```

The uiAction parameter can include the values SIP_SETSIPINFO, SPI_GETSIPINFO, SPI_SETCURRENTIM and SPI_GETCURRENTIM. SIP_SETSIPINFO indicates that pvParam points to a SIPINFO structure (described above). The cbSize, dwImDataSize and pvImDataSize are filled in before calling the SHSipInfo function. In response to this call, the SIPINFO structure is filled in with the current SIP size, state, and visible desktop rectangle. If both dWImDataSize and pvImData are nonzero, the data size and pointer are sent to the Input Method 64. If the Input Method 64 is called but does not provide Input Method-specific data, or the format or size of

the data passed in is not in a format recognized by the Input Method 64, then the SHSipInfo function call fails (returns zero). If the size and format are supported by the Input Method 64, the Input Method 64 fills in the buffer that is pointed to by pvImData with the Input Method-specific data. Typically, an application 29 will set the pvImDataSize to zero and pvImData to NULL.

A uiAction of SPI_SETSIPINFO indicates that pvParam points to a SIPINFO structure. The SIP window 50 size and state are set to the values specified in the SIPINFO structure. Before changing a SIP value, the application 29 should first obtain the current SIP state by calling SHSipInfo with SPI_GETSIPINFO, then change whatever specific SIP state values it wishes to change before making the SPI_SETSIPINFO call. The cbSize field is set to the size of the SIP in the structure, and if both pvImDataSize and pvImData are not zero, the data size and pointer are sent to the Input Method 64. The SHSipInfo call fails if the Input Method 64 is called and does not allow setting Input Method-specific data, or if the format or size of the passed data is not in a format recognized thereby. If a size and format are supported by the Input Method 64, the Input Method 64 uses the data to set Input Method-specific information. Typically, an application will set the pvImDataSize to zero and pvImData to NULL.

SPI_SETCURRENTIM indicates that pvParam points to a CLSID structure which specifies the CLSID of the Input Method 64 to which the SIP will switch. If the CLSID is not valid, or if the specified Input Method 64 cannot be loaded, the call fails (return value equals zero) and a default Input Method 64 (e.g., the QWERTY-like keyboard 66) is loaded.

Lastly, a uiAction of SPI_GETCURRENTIM indicates that pvParam points to a CLSID structure that receives the CLSID of the currently selected Input Method 64.

The IInputMethod Interface

IInputMethod is the interface implemented by the Input Method 64 components. The SIP manager 58 calls the methods of this interface to notify the Input Method 64 of state changes, and request action and information from the Input Method 64. In general, if the called method succeeds, a success is returned, and conversely, if the method fails, a failure result is returned. The following table sets forth the method calls available in this IInputMethod interface:

```
Interface IinputMethod : Iunknown
{
    HRESULT Select( [in] HWND hwndSip );
    HRESULT Deselect( void );
    HRESULT Showing ( void );
    HRESULT Hiding ( void );
    HRESULT GetInfo ( [out] IMINFO *pimi );
    HRESULT ReceiveSipInfo ( [in] SIPINFO *psi );
    HRESULT RegisterCallback ( [in] IIMCallback* pIMCallback );
    HRESULT GetImData ( [in] DWORD dwSize, [out] LPVOID pvImData );
    HRESULT SetImData ( [in] DWORD dwSize, [in] LPVOID pvImData );
    HRESULT UserOptionsDlg ( [in]  HWND hwndParent );
}
```

An Input Method 64 will ordinarily receive a Select(), GetInfo(), ReceiveSipInfo() and Register Callback() method call, in sequence, before rendering the SIP window 50 space or responding to user actions. When the SIP window 50 is displayed (i.e., turned on), Showing() will be called by the SIP manager 58, after which the Input Method 64 issues a WM_PAINT message to render the SIP window 50.

The Select() method is called when the Input Method 64 has been selected into the SIP. The Input Method 64 generally performs any desired initialization in response to this call. The Input Method is responsible for drawing the entire client area of the SIP window 50, and thus ordinarily creates its windows and imagelists (collections of displayable bitmaps such as customized icons) in response to this call. For example, the window handle of the SIP window 50 is provided to

the Input Method 64 as a parameter accompanying this Select() method call, and the Input Method normally creates a child window of this SIP window 50. The Input Method 64 is also provided with a pointer to a value, which is set to nonzero by the Input Method 64 if the method call is successful or zero if not successful.

The Deselect() method is called when the Input Method 64 has been selected out of the SIP. The Input Method's window should be destroyed in response to this call, and the Input Method 64 will typically perform any other cleanup at this time.

The Showing() method will cause the SIP window 50 to be shown upon return from the call. Note that the SIP window 50 is not visible prior to this call, and that once the SIP window 50 is shown, this window and its children will receive paint messages. Conversely, the Hiding() method hides the SIP window 50 upon return from the call. Accordingly, the Showing() and Hiding() methods are used to toggle the SIP window 50 between its open and closed states.

The GetInfo() method is called when the system is requesting information about the Input Method 64. The information requested includes flags indicating any special properties of the Input Method 64, the handles of two imagelists which contain masked bitmaps that are to be

displayed on the SIP button 52 when that Input Method 64 is active, indices into the specified imagelists, and a rectangle indicating the preferred size and placement of the Input Method 64. The call includes a parameter, pimi, which is a pointer to a data structure (IMINFO) that the Input Method 64 should fill in with appropriate data. The call also provides a pointer to a value that the Input Method should set to nonzero to indicate success and zero to indicate failure. More particularly, the IMINFO data structure is represented in the following table:

```
Typedef struct {
     DWORD cbSize;
     HIMAGELIST hImageNarrow;
     HIMAGELIST hImageWide;
     Int iNarrow;
     Int iWide;
     DWORD fdwFlags;
     Rect rcSipRect;
} IMINFO;
```

The cbSize field contains the size of the IMINFO structure, and is filled in by the SIP manager 58 prior to calling calling GetInfo(). The hImageNarrow field is a handle to an imagelist containing narrow (16 x 16) masked bitmaps for the Input Method 64. Similarly, hImageWide is a handle to the imagelist containing wide (32 x 16) masked bitmaps. The SIP manager 58 displays one of the bitmaps (e.g., on the taskbar 56) to indicate the Input Method 64 that is currently

selected. Note that the SIP manager 58 may use the 16 x 16 or 32 x 16 bitmaps at various times depending on how it wishes to display the bitmap.

The iNarrow field is an index into the hImageNarrow imagelist indicating which bitmap of several possible from that (narrow) imagelist should currently be displayed. Similarly, the iWide field is an index into the hImageWide imagelist indicating which bitmap from that (wide) image list should currently be displayed. Note that the Input Method 64 can initiate a change of the bitmap displayed in the SIP taskbar button 52 by calling IIMCallback::SetImages (described below).

The fdwFlags field indicates the visible, docked and locked states (SIPF_ON SIPF_DOCKED and SIPF_LOCKED) of the Input Method 64, as well as any special Input Method flags that may be defined in the future. Note that the SIP state flags are ignored for the GetInfo() method, but are used in the SetImInfo callback method as described below.

Lastly, the rcSipRect field describes the size and placement of the SIP rectangle. The sizing and placement information returned from GetInfo() may be used by the SIP when determining an initial default size and placement. When used, the SetImInfo callback method (described below) specifies the new size and placement of the SIP window 50.

The ReceiveSipInfo() method provides information to the Input Method 64 about the SIP window, including the current size, placement and docked status thereof. This call is made whenever the user, an application 29 or the Input Method 64 changes the SIP state. When the SIP manager 58 sends this information during Input Method initialization, the SIP manger 58 is informing the Input Method 64 of the default SIP settings. The Input Method 64 can choose to ignore these defaults, however the values given are ones that either the user has selected or values that have been recommended as expected or accepted SIP values for that platform. A pointer to the SIPINFO structure that includes this information is passed with this call.

The RegisterCallback method is provided by the SIP manager 58 to pass a callback interface pointer to the Input Method 64. In other words, the RegisterCallback method call passes an IIMCallback interface pointer as a parameter to the Input Method 64, whereby the Input Method 64 can call methods on this interface to send information back to the SIP manager 58 as described below. The Input Method 64 uses the callback interface pointer to send keystrokes to applications 29 via the SIP manager 58 and to change its SIP taskbar button icons 52.

The GetImData() method is called when an application program 29 has asked the SIP for the SIPINFOdata structure and has provided a non-NULL pointer for the pvImData member of the SIPINFO structure. The application 29 will ordinarily cause this call to be made when requesting some special information from the Input Method 64. Two parameters are passed with this call, dwsize, the size of the buffer pointed to by pvImData, and pvImData, a void pointer to a block of data in the application 29.

With this call, the application 29 is essentially requesting that the Input Method 64 fill the block with information, wherein the size and format of the data are defined by the Input Method 64. This call is designed for Input Methods 64 that wish to provide enhanced functionality or information to applications. By way of example, a SIP-aware application may wish to know whether a character was entered by way of the SIP or by some other means. An input method 64 can thus respond to the application's request by filling the block.

The SetImData() method is called when an application 29 has set the SIPINFO data structure and has provided a non-NULL pointer for the pvImData member of the SIPINFO structure. The application 29 will ordinarily cause this call to be made when requesting that the Input Method 64 set some data therein.

The parameters passed with this call include dwsize, the size of the buffer pointed to by pvImData, and pvImData, a void pointer to a block of data in the application 64.

The IIMCallback Interface

The Input Method 64 uses the IIMCallback interface to call methods in the SIP manager 58, primarily to send keystrokes to the current application or to change the icon that the taskbar 56 is displaying in the SIP button 52. The Input Method 64 ordinarily calls the IIMCallback methods only in response to a call thereto which was received through an IInputMethod method call. In general, if the function succeeds, the return value will be a success HRESULT, while conversely, if the function fails, the return value is a failure HRESULT.

The following table represents the IIMCallback Interface:

```
Interface IIMCallback :
Iunknown
{
     Hresult SetImInfo(
               IMINFO *pimi );

     Hresult SendVirtualKey (
               BYTE bVk,
               DWORD dwFlags );

     Hresult SendCharEvents(
               UINT uVk,
               UINT uKeyFlags,
               UINT uChars,
               UINT *puShift,
```

```
                UINT *puChars );

    Hresult SendString(
                BSTR ptrzStr,
                DWORD dwChars );
}
```

The first callback, SetImInfo() is called by the Input Method 64 to change the bitmaps shown on the SIP taskbar button 52 representing the current SIP, or to change the visible/hidden state of the SIP window 50. It is also sent by the Input Method 64 to the SIP manager 58 as a notification when the Input Method 64 has changed the size, placement or docked status of the SIP window 50. By this mechanism, the various Input Methods 64 are able to alert the SIP manager 58 to these types of changes so that the two remain synchronized. By way of example, an Input Method 64 may wish to have a user interface element which allows the user to toggle between a docked state and a floating state, or between one or more subpanels (e.g. keyboard with buttons to switch to a number and/or symbol panel or international symbol panel). The Input Method 64 uses this call to inform the SIP manager 58 of each change in state.

Although not necessary to the invention, all values passed in the IMINFO structure are used by the SIP manager 58. Consequently, the Input Method 64 should first determine the

current state of the SIP window 50 as provided by the SIP manager 58 in the SIPINFO structure received via a prior ReceiveSipInfo() method call, described above. Then, the Input Method 64 should make changes to only those settings in which a change is desired, and pass a full set of values back in the IMINFO structure. The pimi parameter is sent as a pointer to an IMINFO structure representing the new Input Method 64 settings, including the size, placement and state of the SIP window 50 as well as the desired Input Method 64 images.

In response to the SetImInfo() call, the SIP manager 58 will show or hide the SIP window 50 as specified in the fdwFlags of the IMINFO structure. However, the SIP manager 58 will not resize or move the SIP window 50 if requested, but will instead update the size and placement information returned to applications 29 when queried. If the specified values represent a change from the current SIP state, the SIP manager 58 will notify applications 29 that the SIP state has changed via a WM_SETTINGCHANGE message, described above.

The SendVirtualKey() callback is used by an Input Method 64 to simulate a keystroke for a virtual key, e.g., a character or the like entered via the touch screen display 32 or some other Input Method 64. The key event will be sent to the window which currently has focus (i.e., the window which

would have received keyboard input had a key been pressed on an external keyboard). The SendVirtualKey callback modifies the global key state for the virtual key sent, whereby, for example, an Input Method 64 can use this function to send SHIFT, CONTROL, and ALT key-up and key-down events, which will be retrieved correctly when the application 29 calls the GetKeyState() API. The SendVirtualKey callback should be used to send virtual key events that do not have associated characters (i.e., keys that do not cause a WM_CHAR sent as a result of TranslateMessage. Note that WM_CHAR, TranslateMessage and other key-related messages are described in the reference "Programming Windows 95", Charles Petzold, *supra*). If character-producing virtual keys are sent via this function, they will be modified by the global key state. For example, a virtual key of VK_5 that is sent when the shift state is down will result in a '%' WM_CHAR message for certain keyboard layouts.

Parameters sent with this callback include bVk, which is the virtual keycode of the key to simulate, and dwFlags. The dwFlags may be a combination of a SIPKEY_KEYUP flag, (used to generate either a WM_KEYUP or WM_KEYDOWN), a SIPKEY_SILENT flag, (the key press will not make a keyboard click even if clicks are enabled on the device), or zero.

The SendCharEvent callback allows an Input Method 64 to send Unicode characters to the window having focus, while also determining what WM_KEYDOWN and WM_KEYUP messages the application 29 should receive. This allows the Input Method 64 to determine its own keyboard layout, as it can associate any virtual key with any characters and key state. In keeping with one aspect of the invention, applications 29 thus see keys as if they were sent from a keyboard (i.e., they get WM_KEYDOWN, WM_CHAR, and WM_KEYUP messages). Thus, unlike the SendVirtualKey() function, this function does not affect the global key state. By way of example, with the SendCharEvent callback, the Input Method 64 can determine that the shifted (virtual key) VK_C actually sent the Unicode character 0x5564. The shift state flag (specified in the puShift parameter, described below) that is associated with the first character to be sent determines whether a WM_KEYDOWN or WM_KEYUP is generated.

Parameters include uVk, the virtual keycode sent in the WM_KEYUP or WM_KEYDOWN message generated as a result of this function, and a uKeyFlags parameter, a set of KEY state flags that are translated into the lKEYData parameter received in the WM_CHAR, WM_KEYUP or WM_KEYDOWN messages received by the application 29 as a result of this call. Only the KeyStateDownFlag, KeyStatePrevDownFlag, and KeyStateAnyAltFlag

key state flags are translated into the resulting lKeyData parameter. The uChars parameter represents the number of characters corresponding to this key event, while the puShift parameter is a pointer to a buffer containing the corresponding KEY_STATE_FLAGS for each character to be sent. If the KeyStateDownFlag bit is sent, this function generates a WM_KEYDOWN message, otherwise it generates a WM_KEYUP message. Lastly, the puChars parameter is a pointer to a buffer containing the characters to be sent.

An Input Method 64 may use the SendString callback to send an entire string to the window which currently has the focus, whereby a series of WM_CHAR messages are posted to the application 29. An Input Method 64 would typically use this callback after it has determined an entire word or sentence has been entered. For example, a handwriting recognizer or speech recognizer Input Method 64 will use the SendString callback after it has determined that a full word or sentence has been entered.

Parameters of the SendString callback include ptszStr, a pointer to a string buffer containing the string to send, and dwSize, the number of characters to send. This number does not include the null-terminator, which will not be sent.

As can be seen from the foregoing detailed description, there is provided an improved method system for entering user

data into a computer system. The method and system are both efficient and flexible, and function with touch-sensitive input mechanisms. With the system and method, a plurality of applications can receive user input from a common input method, while interchangeable input methods may be selected from among a set thereof for each application. The method and system are cost-effective, reliable, extensible and simple to implement.

While the invention is susceptible to various modifications and alternative constructions, a certain illustrated embodiment thereof is shown in the drawings and has been described above in detail. It should be understood, however, that there is no intention to limit the invention to the specific form disclosed, but on the contrary, the intention is to cover all modifications, alternative constructions, and equivalents falling within the spirit and scope of the invention.

WHAT IS CLAIMED IS:

Sub a1

1. A system for receiving user data input into a computer system, comprising, an operating system, an interface for passing user input data to the operating system, a plurality of input methods, means for selecting one of the input methods as a selected input method, means for receiving user data input via the selected input method, and a communication mechanism for passing information about the received user data to the interface, the interface passing the information to the operating system.

2. The system of claim 1 wherein the operating system comprises a graphical windowing environment, further comprising an input panel window on a touch-sensitive display screen, and wherein the input method includes an input panel and means for drawing the input panel in the input panel window.

3. The system of claim 2 wherein the input panel includes an image representing a keyboard having a plurality of keys thereon, and the means for receiving user data input via the selected input panel includes means for

detecting user activity at screen locations corresponding to the keys of the keyboard.

4. The system of claim 2 further comprising means for selectively displaying and hiding the display of the input panel window.

5. The system of claim 2 wherein interaction with the input panel does not cause the input panel window to receive focus.

Sub a2

6. The system of claim 1 further comprising a touch-sensitive display screen, wherein the means for selecting the selected input method includes means for detecting user interaction with the touch-sensitive display screen.

7. The system of claim 1 further comprising an application program running under the operating system, and wherein the means for selecting the input method includes means for transferring information from the application program to the interface.

8. The system of claim 1 wherein the communication mechanism includes means in the input method for calling

functions to be carried out by the interface, and means in the interface for calling functions to be carried out by the input method.

9. The system of claim 1 wherein the input method comprises a COM object, and the communication mechanism includes means for calling methods in the interface.

10. The system of claim 1 wherein the operating system comprises a graphical windowing environment, further comprising an input panel window on a touch-sensitive display screen, wherein the input method includes an input panel and means for drawing the input panel in the input panel window, and wherein the interface includes means for passing state information corresponding to the state of the input panel window through the communication mechanism to the input method.

11. The system of claim 10 further comprising means for selectively displaying and hiding the display of the input panel window, and wherein the state information includes a flag indicative of the displayed or hidden status of the window.

12. The system of claim 10 wherein the state information includes a flag indicative of the docked or floating status of the window.

13. The system of claim 10 wherein the state information includes the size or position of the window.

Sub A3

14. The system of claim 10 wherein the input method includes a plurality of bitmaps, and wherein the input method includes means for passing information corresponding to a selected one of the bitmaps through the communication mechanism to the interface.

15. The system of claim 14 wherein the interface includes means for displaying the bitmap as an icon on the display screen.

Sub C4

16. ~~In a computer system having a graphical windowing~~ environment for running windows-based applications, a method of receiving user input into the system, comprising the steps of, providing an interface for passing user input to the graphical windowing environment, selecting an input method from a plurality of available input methods, installing the selected input method by connecting the ~~input method to the interface for communication therewith,~~

receiving user input data via the input method, communicating information representative of the user input data to the interface, and passing the information from the interface to the graphical windowing environment.

17. The system of claim 16 further comprising the steps of providing an input panel window on a display screen and drawing an input panel in the input panel window.

18. The method of claim 17 wherein the display screen is a touch-sensitive input device, and wherein the step of selecting the selected input method includes the steps of displaying an icon at a screen location, and detecting a user event at the icon location.

19. The method of claim 17 further comprising the step of toggling between a displayed and hidden state of the input panel window.

20. The method of claim 19 further comprising the step of passing state information corresponding to the displayed or hidden state of the input panel window to the input method.

21. The method of claim 17 further comprising the steps of adjusting the size or position of the input panel window, and passing state information corresponding to the size or position of the input panel window to the input method.

22. The method of claim 17 further comprising the step of toggling between a docked and floating state of the input panel window, and further comprising the step of passing state information corresponding to the docked or floating state of the input panel window to the input method.

23. The method of claim 16 further comprising the step of processing data input to the input method to convert the data to keystroke information.

24. The method of claim 16 wherein the step of communicating information representative of the user input data to the interface includes the step of calling at least one method of the interface.

25. The method of claim 16 wherein the input method includes a plurality of bitmaps, and further comprising the step of passing information corresponding to a selected one of the bitmaps from the input method to the interface.

26. The method of claim 25 further comprising the step of displaying the selected bitmap as an icon on the display screen.

Sub G4

27. A system for receiving user input data into a computer system having a graphical windowing environment, comprising, a touch sensitive display screen for displaying images and detecting user contact or proximity thereto, a management component operatively connected to the graphical windowing environment and including means for creating an input panel window for display thereof by the graphical windowing environment on the screen, a plurality of input methods, each input method including a communication means for calling functions of the management component and further including an input panel corresponding thereto, means for selecting one of the input methods as a selected input method, the selected input method drawing the input panel corresponding thereto in the input panel window, means for receiving user data input via the input panel,

the communication means calling a function of the management component to pass the user data thereto and the management component communicating the user data to the graphical windowing environment.

Sub C5

28. ~~A system for receiving user data input into at~~ least one of a plurality of applications of a computer system, comprising, a management component, a plurality of input methods, each input method for receiving user data when active, wherein one of the plurality of input methods is active and is operatively connected to the management component for passing user data thereto, the management component passing the data to at least one of the ~~applications independent of the input method used~~.

29. The system of claim 28 further comprising an operating system, wherein the management component passes the data to the application program via the operating system.

30. The system of claim 28 wherein the operating system comprises a graphical windowing environment, and further comprising an input panel window on a touch-sensitive display screen.

31. The system of claim 28 wherein the input method converts user input to Unicode characters for passing to the application.

32. The system of claim 28 wherein the management component converts user input to Unicode characters for passing to the application.

33. The system of claim 28 including a mechanism for passing information from the application program to the management component.

34. The system of claim 28 wherein the input method comprises a ~~COM~~ Component Object Model object.

35. The system of claim 30 wherein the input method includes a plurality of bitmaps, and wherein the input method communicates bitmap information to the management component.

36. The system of claim 35 wherein the management component communicates with an operating system to display the bitmap as an icon on the display screen.

Sub a5

37. A method of inputting user data into a mobile computing device to be used by an application, comprising the steps of:

selecting one input method from a plurality of input methods installed on the mobile computing device;

invoking the selected input method within a input panel window displayed by the mobile computing device; and

accepting user data entered in the input panel window in accordance with the selected input method, wherein the entered user data is supplied to the application irrespective of the input method selected.

38. The method of claim 37 wherein the mobile computing device includes a touch-sensitive display, and wherein the step of accepting user data includes the step of detecting user interaction with the touch-sensitive display.

39. The method of claim 37 wherein the input method comprises a ~~COM~~ Component Object Model object, and wherein the step of invoking the selected input method includes the step of instantiating the input method.

B

40. The method of claim 37 further comprising the step of converting the input data to a Unicode character value.

41. ~~A system for receiving user~~ input into a computer system for use with at least one executable application of the computer system, comprising, a plurality of input methods, each input method for accepting input from a user of the computer system, at least one application having current focus on the computer system, a computer operating system designed to accept user input from at least one standard input device and for supplying said user input to the at least one application having focus, and an interface manager operably interfaced with at least one of the plurality of input methods for transforming the input from the user via the at least one input method to correspond to input from said at least one standard input device and operably interfaced with the computer operating system so that the at least one input method is not constrained by the at least one application having focus.

42. The system of claim 41, further comprising a plurality of applications executable on the computer system such ~~that when any one of the applications is focused, said~~

application that is focused is operable to receive input from a user through the operating system from any of the plurality of input methods without modification to said application.

43. The system of claim 41, wherein the standard input device comprises a hardware keyboard.

44. A method of providing a user interface in a computer system for receiving user input for use by an application running on the computer system, comprising the steps of, opening a input panel window on a display of the computer system, selecting one of a plurality of input methods for supplying user input to the computer system, and opening an input method window corresponding to the selected input method within the input panel window wherein a user provides input within the input method window in accordance with the selected input method.

45. The method of claim 44 further comprising the step of providing an input panel button on the display of the computer system, the input panel button being responsive to open and to close the input panel window.

46. The method of claim 44, further comprising the steps of, providing an input method button on the display of the computer system, the input method button being responsive to display a selectable list of the plurality of input methods, receiving a selection of one of the plurality of input methods, and in response, closing an input method window corresponding to an input method other than a selected input method, and opening an input method window corresponding to the selected input method.

Add A6

Add C8

11/B 7-27-00 NP (ALE) Enter

PATENT
Attorney Docket No. 1260

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:
TOEPKE et al.

Group Art Unit: 2774

Serial No.: 08/991,277

Examiner: NGUYEN, K

Filed: December 16, 1997

For: SOFT INPUT PANEL SYSTEM AND METHOD

AMENDMENT AND RESPONSE UNDER 37 CFR 1.116

Assistant Commissioner for Patents
Washington, D.C. 20231

Dear Sir:

This communication is a response to the final Office action mailed May 9, 2000 (the "Office Action"). Entry of the following amendment under the provisions of 37 C.F.R. § 1.116 is respectfully solicited.

In The Claims:

Please amend the claims as follows:

Sub C1 B1 Cont

1. ~~(Twice Amended)~~ ~~A system for receiving user~~ data input into a ~~computer~~ system having an application ~~program, comprising:~~

In re Application of Toepke et al.
Serial No. 08/991,277

~~a plurality of input methods~~, each input method being an interchangeable software component configured to accept the user data input from an input device associated with the computer system,

means for receiving the user data input via a selected input method, and

a management component configured to identify one of the input methods as the selected input method, to load the selected input method into memory, to communicate with the selected input method to identify information about the received user data, and to pass the information about the ~~received user data to the application program~~.

B1 Cancel

In claim 9, please replace "COM" with -Component Object Model-- in both places.

In claim 34, please replace "COM" with -Component Object Model--.

In claim 39, please replace "COM" with -Component Object Model--.

In re Application of Toepke et al.
Serial No. 08/991,277

REMARKS

Claims 1-50 are now pending in the application. By this paper, claims 1, 9, 34 and 39 have been amended to more particularly point out and distinctly claim the subject matter of the invention, and/or to present the rejected claims in better form for consideration on appeal. Applicants submit that the amendments to the claims are unnecessary to distinguish the claimed subject matter from the prior art, and maintain that the claims recited patentable subject matter as-filed and as amended prior to this amendment. The rejections are traversed as explained in the following remarks.

Rejections under Section 112

In the Office Action, claims 9, 34, and 39 were rejected under 35 U.S.C. § 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter of the invention. In particular, the Office Action requested clarification of the acronym "COM." Applicants have amended those claims to specify that COM

means Component Object Model as defined in the specification at page 12, line 3.

Rejections under Section 103(a)

Claims 1-50 were rejected under 35 U.S.C. § 103(a) as being unpatentable in view of the teachings of U.S. Patent No. 5,644,339 to Mori et al. (hereafter "Mori et al.") in combination with the teachings of U.S. Patent No. 5,818,425 to Want et al. (hereafter "Want et al."). Applicants submit that the rejections are improper because the cited art simply does not teach "input methods" within the meaning of the present invention or a management component within the meaning of the present invention. Applicants incorporate by reference the Amendment filed on February 1, 2000.

For the sake of clarity, the independent claims of the application are discussed first in this response. Applicants submit that the independent claims are allowable, and therefore the dependent claims are allowable because they are dependent upon allowable claims. Nevertheless, applicants submit that the dependent claims further define additional subject matter not shown or described in the prior art.

The present invention provides a mechanism by which applications on a computer may make use of common software input methods. The input methods of the present invention are interchangeable software components by which the user provides character, text, or other user data via an input device, such as a touch-screen display, a microphone, or the like. User data input to the computer is received by a selected input method and passes information related to the received user data to a management component. The management component passes the information to an application having input focus. In essence, the input method and management component simulate a standard hardware input device on a portable computer. The invention overcomes limitations in the prior art by allowing multiple applications to share from among the same set of software input methods, thereby eliminating the duplication of similar input methods in each application. Moreover, applications may still provide tailored input methods without the need to incorporate the input method directly in the application.

The Cited References Do Not Describe a Management Component

Claim 1 recites "a management component configured to identify one of the input methods as the selected input method, to load the selected input method into memory, to communicate with the selected input method to identify information about the received user data, and to pass the information about the received user data to the application program." Nowhere is such a component configured in that manner taught or suggested by the cited references.

The Office Action contends that Mori et al. discloses, in Figure 3, a management component configured as described in the claimed invention. Figure 3 of Mori et al. is a flow chart describing the selection and filing of information input as memo information. Mori et al., column 3, lines 44-45. In particular, the flow chart describes a process for identifying a type of character or symbol data being displayed, and appropriately formatting a buffer based on the data type. Applicants are unable to identify anything in Figure 3 that even remotely describes the management component recited in the claims. Management components related to the one recited in claim 1 are also found in

claims 27 and 28. Applicants submit that the rejections of claims 1, 27 and 28 are improper at least because the cited art simply does not disclose, suggest or provide any motivation for a management component as recited in the claims.

The Cited References Do Not Describe Input Methods

Independent claims 1, 16, 27, 28, 37, 41, and 44 include recitations directed at input methods. In the context of the description of the invention, the input methods claimed are interchangeable software components by which the user provides character, text, or other user data via an input device. The Office Action cites Mori et al. (Figures 1-4; column 4, lines 1-47; and column 5, lines 1-38) as disclosing input methods within the context of the claimed invention. The Office Action does not specify what components or teachings of Mori et al. disclose the input methods, and applicants are unable to locate any.

Generally, Mori et al. describe an electronic information apparatus for receiving information input in one format associated with one operating mode (such as a memo mode) and for transforming the information into another

format associated with another operating mode (such as an address note mode). The electronic input apparatus of Mori et al., as described, receives input from a digitizer (element 10 of Figure 1). The input is recognized as characters or symbols and then stored in a video buffer to be displayed. Mori et al. describe a conventional system for recognizing handwriting input and displaying characters based on that input. Nothing in Mori et al. discloses, suggests or provides any motivation for a plurality of input methods as claimed. Thus, applicants submit that each of the independent claims is allowable over the art of record for at least this additional reason.

Independent Patentability of the Dependent Claims

Applicants submit that the dependent claims are allowable for at least the reasons set forth above regarding the independent claims. However, applicants submit that each dependent claim also includes additional patentable distinctions as set forth in part below.

Regarding claim 4, applicants could find nothing in the cited references related to selectively displaying and hiding the display of the input panel window. This aspect of the

invention is important because it allows an increased amount of display screen to be dedicated to display information related to an application when user input is not appropriate. Nowhere do the cited references describe, teach or suggest such an aspect.

Regarding claims 5 and 42, applicants could find nothing in the cited references remotely related to causing an input panel window to receive focus. The concept of input focus is completely absent from the cited references. Nor does the Office Action state its grounds for rejecting claim 5. If the rejection of claim 5 is maintained, applicants respectfully request a clarification of the Office action's allegations.

Regarding claims 8 and 24, nowhere do the cited references describe, teach or suggest a mechanism in an input method for calling functions or methods to be carried out by an interface. As mentioned above, the cited references describe no structure that is equivalent (or even analogous) to the input method of the claimed invention. Likewise, the cited references describe nothing related to a management

component that is capable of carrying out functions called by an input method.

Regarding claims 11, 12, 19, and 20 the Office action contends that Figure 5 of Mori et al. describes state information including "a flag indicative of the docked [state] of the window." Applicants respectfully submit that Figure 5 of Mori et al. is a simulated screen display showing nothing but text. There is nothing in Figure 5 even remotely related to the claimed elements. In addition, the Office Action appears to have withdrawn the previous position that the cited references disclose "a flag indicative of the displayed or hidden status of the window." Applicants respectfully request, if the Office action maintains the assertions, that a specific disclosure of those flags be identified.

Regarding claim 21, applicants could not find, nor does the Office action designate, anything in the cited references related to passing state information associated with an input panel window to an input method. If the rejection of claim 21 is maintained, applicants respectfully request a clarification of the Office action's allegations.

Regarding claim 22, the Office Action appears to have withdrawn the previous contention that Want et al. describe toggling between a docked and floating state of an input panel window. The Office Action now appears to completely overlook this aspect of the claimed invention. Nowhere is there any discussion, mention, or hint that an input panel window may be toggled between a docked and a floating state, as defined in the specification and recited in this claim.

Impermissible Combination of References

Claims 1-50 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Mori et al. in view of Want et al. In making the rejection, the Office action states that "it would have been obvious to one of ordinary skill in the art at the time the invention was made to add an input device hardware keyboard was *[sic]* disclosed by Want et al. because this arrangement would help to operate of the input device." To the extent understood, applicants strongly disagree with this unsupported, broad conclusory statement, as among other reasons, a hardware keyboard is not an input method as defined in the specification and recited in the claims.

Accordingly, applicants respectfully request withdrawal of the § 103(a) rejections of claims 1-50 based on this unsupported and irrelevant contention in the Office action.

Moreover, in order to support an obviousness rejection, by law there must be some teaching, suggestion, or motivation for combining cited references to achieve the claimed invention. See, e.g., *In re Dembiczak*, 175 F.3d 994, 50 U.S.P.Q.2d 1614 (Fed. Cir. 1999) citing *C.R. Bard, Inc. v. M3 Sys., Inc.*, 157 F.3d 1340, 1352, 48 USPQ2d 1225, 1232 (Fed. Cir. 1998) (describing "teaching or suggestion or motivation [to combine]" as an "essential evidentiary component of an obviousness holding"); see also *Id.* citing *In re Fritch*, 972 F.2d 1260, 1265, 23 USPQ2d 1780, 1783 (Fed. Cir. 1992) (examiner can satisfy burden of obviousness in light of combination "only by showing some objective teaching [leading to the combination]"); *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988); *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992); *In re Geiger*, 815 F.2d 686, 688, 2 USPQ2d 1276, 1278 (Fed. Cir. 1987).

The Office action does not indicate any suggestion or motivation in the prior art of record, either explicit or

otherwise, for combining the references in a manner that would achieve the claimed invention, and has failed to meet the requirement of establishing a case of *prima facie* obviousness. Instead, the Office action provides a broad, conclusory statement that is unsupported by the prior art of record and substantially unrelated to the present invention. However, such a broad conclusory statement regarding the teaching of multiple references, standing alone, is not evidence of obviousness. *In re Dembiczak*, 175 F.3d 994, 50 U.S.P.Q.2d 1614 (Fed. Cir. 1999).

Instead, it appears as if the Office action located references that are substantially unrelated to the present invention, in an attempt to piece together applicants' invention using applicants' own teachings. However, (in addition to failing to reach the claims even when impermissibly combined), such a hindsight reconstruction based on applicant's teachings is clearly impermissible by law. For at least these additional reasons, applicants submit that the claims of the present invention are patentable over the prior art of record including Mori et al.

and Want et al., and respectfully requests withdrawal of the §103(a) rejections of the claims based thereon.

CONCLUSION

In view of the foregoing remarks, it is respectfully submitted that claims 1-50 are in good and proper condition for allowance. Entry of the foregoing Amendment and withdrawal of the pending rejections are respectfully solicited under the provisions of 37 C.F.R. § 1.116. If in the opinion of the Examiner a telephone conference would expedite the prosecution of this application, the Examiner is invited to call the undersigned attorney at (425) 653-3520.

Signed at Bellevue, in the County of King, and State of Washington, July 20, 2000.

Respectfully submitted,

Albert S. Michalik

Albert S. Michalik, Reg. No. 37,395
Attorney for Applicants
Michalik & Wylie, PLLC
14645 Bel-Red Road
Suite 103
Bellevue, Washington 98007
(425) 653-3520 (telephone)
(425) 653-3603 (facsimile)

Exhibit J

To

Joint Claim Chart

Attorney Docket No.: D3001.04

PATENT

IN THE UNITED STATES PATENT & TRADEMARK OFFICE

Inventor: Limin Wang, et al.	)	Confirmation No.: 6466
	)	
	)	Customer No.: 000043471
U.S. Serial No.: 11/026,394	)	
	)	
Filed: December 30, 2004	)	
	)	
Con. of U.S. Ser. No. 10/301,290	)	
Filed: November 20, 2002	)	Art Unit: 2613
	)	
	)	Examiner: Shawn S. An
	)	

Title: MACROBLOCK LEVEL ADAPTIVE FRAME/FIELD CODING FOR DIGITAL VIDEO CONTENT

AMENDMENT

Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir,

In response to the Office action mailed on September 6, 2006, having a shortened statutory period of response set to expire on December 6, 2006, a petition for a three month extension of time up to March 7, 2007 being submitted herewith, please amend the above application as follows:

Amendment To The Claims:

1. (Currently amended) A method of encoding a picture in an image sequence, comprising:

dividing said picture into a plurality of macroblocks, each macroblock containing a plurality of blocks ~~smaller portions~~;

generating a plurality of processing blocks, each processing block being generated by grouping said plurality of macroblocks as a processing block, said plurality of macroblocks including a pair of macroblocks or a group of macroblocks; and

selectively encoding at least one of said processing blocks ~~plurality of smaller portions~~ in frame coding mode and at least one of said processing blocks ~~plurality of smaller portions~~ in field coding mode, ~~wherein said encoding is applied to a pair of blocks or a group of blocks,~~

wherein said encoding is performed in a horizontal scanning path or a vertical scanning path.

2. (Currently amended) The method of claim 1, wherein said processing block includes ~~pair of blocks is~~ a pair of macroblocks.

3. (Currently amended) An apparatus for encoding a picture in an image sequence, comprising:

means for dividing said picture into a plurality of macroblocks, each macroblock containing a plurality of blocks ~~smaller portions~~; and

means for generating a plurality of processing blocks, each processing block being generated by grouping said plurality of macroblocks as a processing block, said plurality of macroblocks including a pair of macroblocks or a group of macroblocks;

means for selectively encoding at least one of said processing blocks ~~plurality of smaller portions~~ in frame coding mode and at least one of said processing blocks ~~plurality of smaller portions~~ in field coding mode, ~~wherein said encoding is applied to a pair of blocks, or a group of blocks,~~

wherein said encoding is performed in a horizontal scanning path or a vertical scanning path.

4. (Currently amended) The apparatus of claim 3 wherein said processing block includes ~~pair of blocks is~~ a pair of macroblocks.

5. (Currently amended) A computer-readable medium having stored thereon a plurality of instructions, the plurality of instructions including instructions which, when executed by a processor, cause the processor to perform the steps of a method for encoding a picture in an image sequence, comprising of:

dividing said picture into a plurality of macroblocks, each macroblock containing a plurality of blocks;

generating a plurality of processing blocks, each processing block being generated by grouping said plurality of macroblocks as a processing block, said plurality of macroblocks including a pair of macroblocks or a group of macroblocks; and

selectively encoding at least one of said processing blocks ~~plurality of smaller portions~~ in frame coding mode and at least one of said processing blocks ~~plurality of smaller portions~~ in field coding mode, ~~wherein said encoding is applied to a pair of blocks or a group of blocks~~,

wherein said encoding is performed in a horizontal scanning path or a vertical scanning path.

6. (Currently amended) A method of decoding an encoded picture having a plurality of processing blocks, each processing block containing macroblocks, each macroblock containing a plurality of blocks, ~~smaller portions~~ from a bitstream, comprising:

decoding at least one of said plurality of processing blocks ~~smaller portions~~ in frame coding mode and at least one of said plurality of processing blocks ~~smaller portions~~ in field coding mode, ~~wherein said decoding is applied to a pair of blocks, or a group of blocks,~~

wherein said decoding is performed in a horizontal scanning path or a vertical scanning path; and using said plurality of decoded processing blocks ~~smaller portions~~ to construct a decoded picture.

7. (Currently amended) The method of claim 6, wherein said processing blocks include ~~pair of blocks is~~ a pair of macroblocks.

8. (Currently amended) The method of claim 6, wherein said decoding is applied to a group of at least four macroblocks ~~blocks~~.

9. (Currently amended) The method of claim 8, further comprising: joining said group of at least four macroblocks ~~blocks~~ into a top field block and a bottom field block when said group of at least four macroblocks ~~blocks~~ is decoded in said field coding mode.

10. (Currently amended) An apparatus for decoding an encoded picture from a bitstream, comprising:

means for decoding at least one of a plurality of processing blocks, each processing block containing macroblocks, each macroblock containing a plurality of blocks, ~~smaller portions~~ from said encoded picture that is encoded in frame coding mode and at least one of said plurality of processing blocks ~~smaller portions~~ that is encoded in field coding mode, ~~wherein said decoding is applied to a pair of blocks, or a group of blocks,~~

wherein said decoding is performed in a horizontal scanning path or a vertical scanning path; and

means for using said plurality of decoded processing blocks ~~smaller portions~~ to construct a decoded picture.

11. (Currently amended) The apparatus of claim 10, wherein said processing blocks include ~~pair of blocks is~~ a pair of macroblocks.

12. (Currently amended) The apparatus of claim 10, wherein said decoding is applied to a group of at least four macroblocks ~~blocks~~.

13. (Currently amended) The apparatus of claim 12, further comprising: means for joining said group of at least four macroblocks ~~blocks~~ into a top field block and a bottom field block when said group of at least four macroblocks ~~blocks~~ is decoded in said field coding mode.

14. (Currently amended) A computer-readable medium having stored thereon a plurality of instructions, the plurality of instructions including instructions which, when executed by a processor, cause the processor to perform the steps of a method for decoding an encoded picture from a bitstream, comprising of:

decoding at least one of a plurality of processing blocks, each processing block containing macroblocks, each macroblock containing a plurality of blocks, ~~smaller portions~~ from said encoded picture that is encoded in frame coding mode and at least one of said plurality of processing blocks ~~smaller portions~~ that is encoded in field coding mode, ~~wherein said decoding is applied to a pair of blocks, or a group of blocks,~~

wherein said decoding is performed in a horizontal scanning path or a vertical scanning path; and

using said plurality of decoded smaller portions to construct a decoded picture.

15. (Currently amended) A bitstream comprising: a picture that has been divided into a plurality of processing blocks, each processing block containing macroblocks, each macroblock containing a plurality of blocks, wherein at least one of said plurality of processing blocks ~~smaller portions~~ from said picture is encoded in frame coding mode and at least one of said plurality of processing blocks ~~smaller portions~~ is encoded in field coding mode, wherein said encoding is performed in a horizontal scanning path or a vertical scanning path.

16. (New) The method of claim 1, wherein each pair of blocks of said image is coded from left to right and from top to bottom if said encoding is performed in said horizontal scanning path, and

wherein each pair of blocks of said image is coded from top to bottom and from left to right if said encoding is performed in said vertical scanning path.

17. (New) The method of claim 1, wherein said pair of macroblocks comprises a top block and a bottom block, where said top block is encoded prior to said bottom block in said frame coding mode.

18. (New) The method of claim 1, further comprising:

splitting said pair of macroblocks into a top field block and a bottom field block when said pair of blocks are encoded in said field coding mode, and where said top field block is encoded prior to said bottom field block.

19. (New) The method of claim 1, wherein said processing block includes a group of at least four macroblocks blocks.

20 (New) The apparatus of claim 3, wherein each pair of blocks of said image is coded from left to right and from top to bottom if said encoding is performed in said horizontal scanning path, and

wherein each pair of blocks of said image is coded from top to bottom and from left to right if said encoding is performed in said vertical scanning path.

21. (New) The apparatus of claim 3, wherein said pair of macroblocks comprises a top block and a bottom block, where said top block is encoded prior to said bottom block in said frame coding mode.

22. (New) The apparatus of claim 3, wherein said pair of macroblocks are split into a top field block and a bottom field block when said pair of blocks are encoded in said field coding mode, and where said top field block is encoded prior to said bottom field block.

23. (New) The apparatus of claim 3, wherein said processing block includes a group of at least four macroblocks blocks.

24. (New) The method of claim 6, wherein each pair of blocks of said image is coded from left to right and from top to bottom if said encoding is performed in said horizontal scanning path, and

wherein each pair of blocks of said image is coded from top to bottom and from left to right if said encoding is performed in said vertical scanning path.

25. (New) The method of claim 6, wherein said pair of macroblocks comprises a top block and a bottom block, where said top block is encoded prior to said bottom block in said frame coding mode.

26. (New) The method of claim 6, further comprising:

splitting said pair of macroblocks into a top field block and a bottom field block when said pair of blocks are encoded in said field coding mode, and where said top field block is encoded prior to said bottom field block.

27. (New) The method of claim 6, wherein said processing block includes a group of at least four macroblocks blocks.

28. (New) The apparatus of claim 10, wherein each pair of blocks of said image is coded from left to right and from top to bottom if said encoding is performed in said horizontal scanning path, and

wherein each pair of blocks of said image is coded from top to bottom and from left to right if said encoding is performed in said vertical scanning path.

29. (New) The apparatus of claim 10, wherein said pair of macroblocks comprises a top block and a bottom block, where said top block is encoded prior to said bottom block in said frame coding mode.

30. (New) The apparatus of claim 10, wherein said pair of macroblocks are split into a top field block and a bottom field block when said pair of blocks are encoded in said field coding mode, and where said top field block is encoded prior to said bottom field block.

31. (New) The apparatus of claim 10, wherein said processing block includes a group of at least four macroblocks blocks.

REMARKS

I. Introduction

Claims 1-31 are pending in the above application.

Claims 1-5 and 15 stand rejected under 35 U.S.C. § 102.

Claims 5-14 stand rejected under 35 U.S.C. § 103.

II. Amendments

Claims 1-15 have been amended to more particularly point out what Applicant regards as the inventions therein. Support for the amendments may be found at least in paragraphs 64 and 65 of applicant's disclosure.

Claim 16-31 are newly added.

No new matter has been added.

III. Prior Art Rejections

A. Claims 1-4 and 15 stand rejected under 35 U.S.C. § 102 as being anticipated by Obikane et al. (U.S. Pat. No. 5,504,530).

Anticipation under 35 U.S.C. § 102 requires that each and every element of the claim be disclosed in a prior art reference as arranged in the claim. See, *IPXL Holdings, L.L.C. v. Amazon.com, Inc.*, 430 F.3d 1377, 1380 (Fed. Cir. June 2006) ("a claim is anticipated under 35 U.S.C. § 102 'if each and every limitation is found either expressly or inherently in a single prior art reference'" citing, *Bristol-Myers Squibb Co. v. Ben Venue Labs, Inc.*, 246 F.3d 1368, 1374 (Fed. Cir. 2001). See also, *Akzo N.V. v. U.S. Int'l Trade Commission*, 808 F.2d 1471 (Fed. Cir. 1986); *Connell v. Sears, Roebuck & Co.*, 220 USPQ 193, 198 (Fed. Cir. 1983).

Obikane does not disclose dividing a picture into a plurality of macroblocks, each macroblock containing a plurality of blocks; generating a plurality of processing blocks, each processing block including a plurality of macroblocks including a pair of macroblocks or a group of macroblocks; and selectively encoding at least one of said processing blocks in frame coding mode and at least one of said processing blocks in field coding mode, wherein said encoding is performed in a horizontal scanning path or a vertical scanning path, as substantially recited by the above claims. Obikane discloses an MPEG-2 approach in which encoding and decoding are performed on a macroblock basis. The citations indicated by the Examiner (col. 3: 10-30) merely discuss the formation of a macroblock as having a plurality of smaller blocks. Obikane does not disclose to encoding and decoding processes involving more than one macroblock and does not disclose a scanning path to encode or decode plural macroblocks.

Hence, Obikane does not disclose each and every limitation of amended claims 1, 3 or 15, nor claims 2 and 4 which depend on claims 1 and 3, respectively.

B. Claims 5-8, 10-12 and 14 stand rejected under 35 U.S.C. § 103 as being unpatentable over Obikane.

Obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art. *Ecolochem Inc. v. Southern California Edison Co.*, 227 F.3rd 1361, 56 U.S.P.Q.2d (BNA) 1065 (Fed. Cir. 2000); *In re Dembiczak*, 175 F.3d 994, 999, 50 U.S.P.Q.2D (BNA) 1614, 1617 (Fed. Cir. 1999); *In re Jones*, 958 F.2d 347, 21

U.S.P.Q.2d 1941 (Fed. Cir. 1992); and *In re Fine*, 837 F.2d 1071, 5 U.S.P.Q.2d 1596 (Fed. Cir. 1988). See also MPEP 2143.01.

As discussed above Obikane does not disclose or suggest encoding and decoding processes involving more than one macroblock and does not disclose a scanning path to encode or decode plural macroblocks. Hence, Obikane does not disclose all of the claimed limitation of amended claims 5, 6, 10 or 14, nor claims 7-8 and 11-12 which depend on claims 6 and 10, respectively.

C. Claims 9 and 13 stand rejected under 35 U.S.C. § 103 as being unpatentable over Obikane in view of Mishima et al. (U.S. Pat. No. 5,825,419).

Neither Obikane nor Mishima, taken alone or in combination, disclose or suggest encoding and decoding processes involving more than one macroblock and does not disclose or suggest a scanning path to encode or decode plural macroblocks. Obikane does not disclose such as discussed above. Mishima also does not disclose such and the Examiner does not appear to rely on Mishima as disclosing such.

Hence, neither Obikane nor Mishima, taken alone or in combination disclose or suggest all of the claimed limitation of amended claims 5 or 10 from which claims 9 and 13 depend, respectively.

IV. New Claims

The newly added claims depend on the previously examined independent claims and are believed to be allowable at least for the same reason as their respective base independent claims.

V. Conclusion

Having fully responded to the Office action, the above application is believed to be in condition for allowance. Should any issues arise that prevent early allowance of the above application, the examiner is invited contact the undersigned to resolve such issues.

To the extent an extension of time is needed for consideration of this paper, Applicant hereby request such extension and, the Commissioner is hereby authorized to charge deposit account number 502117 for any fees associated therewith.

Respectfully submitted,

By: //Larry T. Cullen//
Larry T. Cullen
Reg. No.: 44,489

Motorola Connected Home Solutions
101 Tournament Drive
Horsham, PA 19044
(215) 323-1797

Notice of Allowability	Application No.	Applicant(s)	
	11/026,394	WANG ET AL.	
	Examiner	Art Unit	
	Shawn S. An	2621	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to *3/6/07*.

2. ☒ The allowed claim(s) is/are *1-31*.

3. ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
   a) ☐ All b) ☐ Some* c) ☐ None of the:
      1. ☐ Certified copies of the priority documents have been received.
      2. ☐ Certified copies of the priority documents have been received in Application No. _____ .
      3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

   * Certified copies not received: _____.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application.
THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.

4. ☐ A SUBSTITUTE OATH OR DECLARATION must be submitted. Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL PATENT APPLICATION (PTO-152) which gives reason(s) why the oath or declaration is deficient.

5. ☐ CORRECTED DRAWINGS (as "replacement sheets") must be submitted.
   (a) ☐ including changes required by the Notice of Draftsperson's Patent Drawing Review (PTO-948) attached
      1) ☐ hereto or 2) ☐ to Paper No./Mail Date _____.
   (b) ☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date _____.

   Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).

6. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1. ☐ Notice of References Cited (PTO-892)
2. ☐ Notice of Draftperson's Patent Drawing Review (PTO-948)
3. ☒ Information Disclosure Statements (PTO/SB/08), Paper No./Mail Date _____
4. ☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material
5. ☐ Notice of Informal Patent Application
6. ☐ Interview Summary (PTO-413), Paper No./Mail Date _____ .
7. ☒ Examiner's Amendment/Comment
8. ☒ Examiner's Statement of Reasons for Allowance
9. ☐ Other _____.

EXAMINER'S AMENDMENT

1. An Examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to Applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

IN THE CLAIMS:

A) Please amend claims 1, 3, 5-6, 10, 14-18, 20-22, 24-26, and 28-30 as follows:

1. (Currently Amended) A method of encoding a picture in an image sequence, comprising:

dividing said picture into a plurality of macroblocks, each macroblock containing a plurality of blocks;

generating a plurality of processing blocks, each processing block being generated by grouping said plurality of macroblocks as a processing block, said plurality of macroblocks including a pair of macroblocks or a group of macroblocks; and

selectively encoding at least one of said processing blocks at a time in frame coding mode and at least one of said processing blocks at a time in field coding mode,

wherein said encoding is performed in a horizontal scanning path or a vertical scanning path.

3. (Currently Amended) An apparatus for encoding a picture in an image sequence, comprising:

means for dividing said picture into a plurality of macroblocks, each macroblock containing a plurality of blocks; and

means for generating a plurality of processing blocks, each processing block being generated by grouping said plurality of macroblocks as a processing block, said plurality of macroblocks including a pair of macroblocks or a group of macroblocks;

means for selectively encoding at least one of said processing blocks at a time in frame coding mode and at least one of said processing blocks at a time in field coding mode,

wherein said encoding is performed in a horizontal scanning path or a vertical scanning path.

5. (Currently Amended) A computer-readable medium encoded with computer executable instructions ~~having stored thereon a plurality of instructions~~, the ~~plurality of~~ computer executable instructions including instructions which, when executed by a processor, cause the processor to perform the steps of a method for encoding a picture in an image sequence, comprising the steps of:

dividing said picture into a plurality of macroblocks, each macroblock containing a plurality of blocks;

generating a plurality of processing blocks, each processing block being generated by grouping said plurality of macroblocks as a processing block, said plurality of macroblocks including a pair of macroblocks or a group of macroblocks; and

selectively encoding at least one of said processing blocks at a time in frame coding mode and at least one of said processing blocks at a time in field coding mode,

wherein said encoding is performed in a horizontal scanning path or a vertical scanning path.

6. (Currently Amended) A method of decoding an encoded picture having a plurality of processing blocks, each processing block containing macroblocks, each macroblock containing a plurality of blocks, from a bitstream, comprising:

decoding at least one of ~~said~~ a plurality of processing blocks at a time, wherein each of said plurality of processing blocks includes a pair of macroblocks or a group of macroblocks, in frame coding mode and at least one of said plurality of processing

blocks at a time in field coding mode, wherein said decoding is applied to a pair of blocks, or a group of blocks,

wherein said decoding is performed in a horizontal scanning path or a vertical scanning path; and

using said plurality of decoded processing blocks to construct a decoded picture.

10. (Currently Amended) An apparatus for decoding an encoded picture from a bitstream, comprising:

means for decoding at least one of ~~said~~ a plurality of processing blocks at a time, each processing block containing a pair of macroblocks or a group of macroblocks, each macroblock containing a plurality of blocks, from said encoded picture that is encoded in frame coding mode and at least one of said plurality of processing blocks at a time that is encoded in field coding mode,

wherein said decoding is performed in a horizontal scanning path or a vertical scanning path; and

means for using said plurality of decoded processing blocks to construct a decoded picture.

14. (Currently Amended) A computer-readable medium encoded with computer executable instructions ~~having stored thereon a plurality of instructions~~, the ~~plurality of~~ computer executable instructions including instructions which, when executed by a processor, cause the processor to perform the steps of a method for encoding a picture in an image sequence, comprising the steps of:

decoding at least one of a plurality of processing blocks at a time, each processing block containing a pair of macroblocks or a group of macroblocks, each macroblock containing a plurality of blocks, from said encoded picture that is encoded in frame coding mode and at least one of said plurality of processing blocks at a time that is encoded in field coding mode,

wherein said decoding is performed in a horizontal scanning path or a vertical scanning path; and

using said plurality of decoded processing blocks to construct a decoded picture.

15. (Currently Amended) A bitstream comprising:

a picture that has been divided into a plurality of processing blocks, each processing block containing a pair of macroblocks or a group of macroblocks, each macroblock containing a plurality of blocks,

wherein at least one of said plurality of processing blocks from said picture is encoded in frame coding mode at a time and at least one of said plurality of processing blocks is encoded in field coding mode at a time,

wherein said encoding is performed in a horizontal scanning path or a vertical scanning path.

16. (Currently Amended) The method of claim 2 ~~1~~, wherein each pair of macroblocks ~~blocks~~ of said image is encoded ~~coded~~ from left to right and from top to bottom if said encoding is performed in said horizontal scanning path, and

wherein each pair of macroblocks ~~blocks~~ of said image is encoded ~~coded~~ from top to bottom and from left to right if said encoding is performed in said vertical scanning path.

17. (Currently Amended) The method of claim 2 ~~1~~, wherein said pair of macroblocks comprises a top block and a bottom block, where said top block is encoded prior to said bottom block in said frame coding mode.

18. (Currently Amended) The method of claim 2 ~~1~~, further comprising:

splitting said pair of macroblocks into a top field block and a bottom field block when said pair of macroblocks ~~blocks~~ are encoded in said field coding mode, and where said top field block is encoded prior to said bottom field block.

20. (Currently Amended) The apparatus of claim 4 ~~3~~, wherein each pair of macroblocks ~~blocks~~ of said image is encoded ~~coded~~ from left to right and from top to bottom if said encoding is performed in said horizontal scanning path, and

wherein each pair of macroblocks ~~blocks~~ of said image is encoded ~~coded~~ from top to bottom and from left to right if said encoding is performed in said vertical scanning path.

21. (Currently Amended) The apparatus of claim 4 ~~3~~, wherein said pair of macroblocks comprises a top block and a bottom block, where said top block is encoded prior to said bottom block in said frame coding mode.

22. (Currently Amended) The appartus of claim 4 ~~3~~, wherein said pair of macroblocks are split into a top field block and a bottom field block when said pair of macroblocks ~~blocks~~ are encoded in said field coding mode, and where said top field block is encoded prior to said bottom field block.

24. (Currently Amended) The method of claim 7 ~~6~~, wherein each pair of macroblocks ~~blocks~~ of said image is decoded ~~coded~~ from left to right and from top to bottom if said decoding ~~encoding~~ is performed in said horizontal scanning path, and

wherein each pair of macroblocks ~~blocks~~ of said image is decoded ~~coded~~ from top to bottom and from left to right if said decoding ~~encoding~~ is performed in said vertical scanning path.

25. (Currently Amended) The method of claim 7 ~~6~~, wherein said pair of macroblocks comprises a top block and a bottom block, where said top block is decoded ~~encoded~~ prior to said bottom block in said frame coding mode.

26. (Currently Amended) The method of claim 7 ~~6~~, wherein said pair of macroblocks is represented by a top field block and a bottom field block in said field coding mode, the method further comprising:

decoding said top field block and said bottom field block, and

joining said top field block and said bottom field block into said pair of macroblocks

~~splitting said pair of macroblocks into a top field block and a bottom field block when said pair of blocks are encoded in said field coding mode, and where said top field block is encoded prior to said bottom field block~~.

28. (Currently Amended) The apparatus of claim 11 ~~10~~, wherein each pair of macroblocks ~~blocks~~ of said image is decoded ~~coded~~ from left to right and from top to bottom if said decoding ~~encoding~~ is performed in said horizontal scanning path, and

wherein each pair of macroblocks ~~blocks~~ of said image is decoded ~~coded~~ from top to bottom and from left to right if said decoding ~~encoding~~ is performed in said vertical scanning path.

29. (Currently Amended) The apparatus of claim 11 ~~10~~, wherein said pair of macroblocks comprises a top block and a bottom block, where said top block is decoded ~~encoded~~ prior to said bottom block in said frame coding mode.

30. (Currently Amended) The apparatus of claim 11 ~~10~~, wherein said pair of macroblocks is represented by a top field block and a bottom field block in said field coding mode, the method further comprising:

decoding said top field block and said bottom field block, and

joining said top field block and said bottom field block into said pair of macroblocks

~~wherein said pair of macroblocks are split into a top field block and a bottom field block when said pair of blocks are encoded in said field coding mode, and where said top field block is encoded prior to said bottom field block~~.

REMARKS:

Claims 1, 3, 5-6, 10, 14-18, 20-22, 24-26, and 28-30 have been amended as discussed above, as authorized by Applicant's attorney, Larry T. Cullen (44,489) on May 24, 2007.

2. Any inquiry concerning this communication or earlier communications from the Examiner should be directed to *Shawn S. An* whose telephone number is 571-272-7324.

SHAWN AN
PRIMARY EXAMINER

5/24/07

Reasons for Allowance

1. As per Applicant's instructions as filed on 3/06/07, claims 1-15 have been amended, and claims 16-31 have been newly added.

2. Claims 1-31 are allowed.

3. Claims 1-31 are allowed after entering the Examiner's Amendment as discussed in the EXAMINER'S AMENDMENT section.

4. Claims 1-31 as amended are allowed as having incorporated novel features of an encoder/decoder comprising:

means/steps for dividing said picture into a plurality of macroblocks, each macroblock containing a plurality of blocks; and

means/steps for generating a plurality of processing blocks, each processing block being generated by grouping said plurality of macroblocks as a processing block, the plurality of macroblocks including a pair of macroblocks or a group of macroblocks;

means/steps for selectively encoding at least one of said processing blocks ***at a time*** *in frame coding mode* and at least one of said processing blocks ***at a time*** *in field coding mode*,

wherein the encoding is performed in a horizontal scanning path or a vertical scanning path.

The prior art of record fails to anticipate or make obvious the novel features (emphasis added on *underlined claim(s) limitations*) as above.

Any comments considered necessary by Applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

Conclusion

5. Any inquiry concerning this communication or earlier communications from the Examiner should be directed to *Shawn S. An* whose telephone number is 571-272-7324.

6. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

7. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

SHAWN AN
PRIMARY EXAMINER

5/24/07